An ADMM Algorithm for Solving ℓ_1 Regularized MPC

Mariette Annergren*, Anders Hansson**, and Bo Wahlberg*

***Abstract*— We present an Alternating Direction Method of Multipliers (ADMM) algorithm for solving optimization problems with an ℓ_1 regularized least-squares cost function subject to recursive equality constraints. The considered optimization problem has applications in control, for example in ℓ_1 regularized MPC. The ADMM algorithm is easy to implement, converges fast to a solution of moderate accuracy, and enables separation of the optimization problem into sub-problems that may be solved in parallel. We show that the most costly step of the proposed ADMM algorithm is equivalent to solving an LQ regulator problem with an extra linear term in the cost function, a problem that can be solved efficiently using a Riccati recursion. We apply the ADMM algorithm to an example of ℓ_1 regularized MPC. The numerical examples confirm fast convergence to moderate accuracy and a linear complexity in the MPC prediction horizon.**

I. Introduction

In this paper we consider optimization problems with an ℓ_1 regularized least-squares cost function subject to recursive equality constraints. This has applications in control. The least squares part is standard in this context and penalizes deviations of the states from the set-point at the same time as keeping the control signal small. The ℓ_1-norm regularization of the cost function promotes sparse solutions, i.e. a solution with many zero entries, [1]. The cost function is known as LASSO, [2]. LASSO is a well-known method in statistics and machine learning, and it has gained a lot of interest in other research communities as well, e.g. system identification, [3].

We propose to solve the optimization problem using an algorithm called Alternating Direction Method of Multipliers (ADMM). ADMM is a special case of Douglas-Rachford splitting, [4], and it is related to other optimization algorithms, e.g. method of multipliers and Bregman iterative algorithms for ℓ_1 problems, [5], [6]. For an overview of ADMM, we refer the reader to [7].

The most costly step in the proposed ADMM algorithm is the projection of an iterate to a set describing a feasible solution. We will show that this projection is equivalent to solving a Linear Quadratic (LQ) regulator problem with an additional linear term in the cost function. This problem can be solved efficiently using a Riccati recursion just as in [8].

We will apply ADMM to the recently introduced ℓ_1 regularized Model Predictive Control (MPC), [9]. The ℓ_1 regularized MPC has an ℓ_1 regularized least-squares cost function. The motivation for ℓ_1 regularized MPC is the reduced actuator activity obtained when using ℓ_1-norm penalty on changes of the input signal [9]. A detailed stability analysis of the closed loop system with ℓ_1 regularized MPC, and results confirming sparse solutions are given in [9]. In ℓ_1 regularized MPC, an optimization problem such as the one we consider is solved at each sampling instant. Hence, the sampling time puts an upper bound on the time that the optimization is allowed to take, and therefore efficient algorithms are needed. It is believed that ADMM is a preferred algorithm for this application based on the result for LASSO, [7]. We will see that this expectation is confirmed in numerical experiments.

Also for traditional MPC an optimization problem has to be solved at each sampling instant, [10]. Because of this many different tailored optimization schemes have been developed to meet the real time requirements of MPC. Typically the optimization problem is a Quadratic Program (QP). There are mainly two different approaches that have been taken. One approach is to compute an explicit off-line solution to the QP which is stored in a look-up table, [11]. This facilitates very fast sampling, but is only feasible for small scale problems. The other approach is to compute the solution on-line as we propose, which is feasible also for medium- and large-scale problems. Among these approaches one can distinguish three different classes of methods: 1) Interior Point (IP) methods, [12], 2) Active-Set (AS) methods, [8] and 3) Fast Gradient (FG) methods, [13]. Riccati recursions play an important role also in IP and AS methods for MPC, [12], [8], since they can be used for these methods to efficiently factorize the matrix involved in the linear system of equations for the search directions. So far they have not been used for FG methods. For IP methods the Riccati recursion has to be re-computed for each iterate of the method. For AS methods it has to be updated, i.e. parts of the old solution can be reused but has to be modified. For ADMM it is possible to use the same Riccati recursion for all iterates. Computing the Riccati recursion, i.e. factorizing the matrix for the search directions, is the most time-consuming task for all these methods. However, the convergence performance is not the same for the different methods, i.e. it takes a different amount of iterations to reach a solution of satisfactory accuracy. For IP methods the number of iterations is typically 20–50 to reach very high accuracy. For active set methods the number of iterations are typically higher, however by considering gradient projection methods on the dual problem speed can be gained, [14], and similar results as for IP methods can be obtained. For fast gradient methods it has in [13] been shown how the number of iterates can be upper bounded to achieve a desired

This work was partially supported by the Swedish Research Council and the Linnaeus Center ACCESS at KTH and the European Research Council under the advanced grant LEARN, contract 267381.

*Automatic Control Lab and ACCESS, School of Electrical Engineering, KTH, SE-100 44 Stockholm, Sweden. (e-mail: {mariette.annergren, bo.wahlberg}@ee.kth.se)

**Division of Automatic Control, Department of Electrical Engineering, Linköpings Universitet, SE-581 83 Linköping, Sweden. (e-mail: anders.g.hansson@liu.se). This work was carried out when the author was a Visiting Professor at University of California, Los Angeles.

accuracy. Other recent relevant publications in relation to efficient methods for MPC include among others [15], [16], [17], [18] and the references therein.

II. Control problem

We consider an open-loop control problem of finding an input sequence that minimizes a finite-horizon cost function, given a model and an initial state. The problem is formulated as follows

$$\begin{array}{ll} \text{minimize} & \|x_H\|_{2,Q}^2 + \sum_{i=1}^H \|y_{i-1}\|_2^2 + \lambda \sum_{i=1}^H \|z_{i-1}\|_1, \\ \text{subject to} & x_i = Ax_{i-1} + Bu_{i-1},\ i = 1,\ldots,H, \\ & y_{i-1} = Cx_{i-1} + Du_{i-1},\ i = 1,\ldots,H, \\ & z_{i-1} = Ex_{i-1} + Fu_{i-1},\ i = 1,\ldots,H, \end{array} \tag{1}$$

where $x_i \in \mathbf{R}^n$ is the state vector, $u_i \in \mathbf{R}^l$ is the input vector, $y_i \in \mathbf{R}^m$ and $z_i \in \mathbf{R}^p$ are auxiliary variables, and where $\|x\|_{2,A}^2 = x^T Ax$. Formulation (1) captures the optimization problems that may occur in ℓ_1 regularized MPC. For example, we can replace the input vector with the change of the input by augmenting the state vector and modifying the system matrices accordingly, see [10].

III. Alternating direction method of multipliers (ADMM)

In this section, we provide a description of the key elements of ADMM. The description is a condensed version of the ones found in [19] and [7]. For a more rigorous overview, we refer the reader to [7].

A. Optimization problem

ADMM is a numerical algorithm for solving optimization problems such as

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & x \in \mathscr{C}, \end{array} \tag{2}$$

for some vector variable $x \in \mathbf{R}^n$, where $f(x)$ is a convex function and $\mathscr{C}$ is a convex set. An equivalent problem to (2) is

$$\begin{array}{ll} \text{minimize} & f(x) + I_{\mathscr{C}}(x_c), \\ \text{subject to} & x = x_c, \end{array} \tag{3}$$

where $I_{\mathscr{C}}(x_c)$ is the indicator function of $\mathscr{C}$, [1].

B. Augmented Lagrangian

The augmented Lagrangian of optimization problem (3) is defined as

$$L_\rho(x, x_c, x_d) = f(x) + I_{\mathscr{C}}(x_c) + (\rho/2)\|x - x_c + x_d\|_2^2, \tag{4}$$

where x_d is the dual variable corresponding to the equality constraint $x = x_c$ scaled by $1/\rho$, and $\rho > 0$ is a tunable parameter. There is no simple way of finding the optimal ρ, however there are guidelines in [7].

C. ADMM steps

The ADMM algorithm consists of three main steps at each iteration k. The three steps are

$$x^{k+1} := \arg\min_x \{f(x) + (\rho/2)\|x - x_c^k + x_d^k\|_2^2\} \tag{5}$$

$$x_c^{k+1} := \Pi_{\mathscr{C}}(x^{k+1} + x_d^k), \tag{6}$$

$$x_d^{k+1} := x_d^k + (x^{k+1} - x_c^{k+1}), \tag{7}$$

where $\Pi_{\mathscr{C}}(x)$ denotes the Euclidean projection of a vector x onto a set $\mathscr{C}$. In the first step (5), we minimize the augmented Lagrangian (4) with respect to x, keeping x_c and x_d fixed. In the second step (6), we minimize the augmented Lagrangian (4) with respect to x_c, keeping x and x_d fixed. In the third and last step (7), we update the scaled dual variable x_d. We then repeat all three steps until convergence. For more details and a complete convergence analysis, we refer the reader to [7].

D. Stopping criteria

The ADMM algorithm is iterated until some stopping criteria are fulfilled. We use criteria based on the primal and dual residuals of the optimization problem. The primal and dual residuals of (3) are

$$e_p^k = (x^k - x_c^k), \quad e_d^k = -\rho(x_c^k - x_c^{k-1}).$$

We terminate the algorithm when

$$\begin{array}{l} \|e_p^k\|_2 \le \sqrt{n}\varepsilon^{\text{abs}} + \varepsilon^{\text{rel}} \max\{\|x^k\|_2, \|x_c^k\|_2\}, \\ \|e_d^k\|_2 \le \sqrt{n}\varepsilon^{\text{abs}} + \varepsilon^{\text{rel}} \rho \|x_d^k\|_2, \end{array} \tag{8}$$

where $\varepsilon^{\text{abs}} > 0$ and $\varepsilon^{\text{rel}} > 0$ are absolute and relative tolerances, respectively. For more details, see [7].

E. Over-relaxation

We may use over-relaxation to improve convergence of the ADMM algorithm, [4], [20], [21]. When using over-relaxation we modify the update x^{k+1} with

$$\hat{x}^{k+1} = \alpha x^{k+1} + (1 - \alpha)x_c^k,$$

where $1.5 \le \alpha \le 1.8$, in the second and the third ADMM steps, (6) and (7). For more details, see [7].

IV. Problem formulation and method

In this section, we describe how the considered optimization problem in (1) can be solved using ADMM.

A. ADMM formulation

The optimization problem in (1) is on the same form as the optimization problem in (2). The vector variables are

$$\begin{array}{ll} x = (x_0, \ldots, x_H), & y = (y_0, \ldots, y_{H-1}), \\ u = (u_0, \ldots, u_{H-1}), & z = (z_0, \ldots, z_{H-1}), \end{array}$$

the objective function is

$$f(x, y, u, z) = \|x_H\|_{2,Q}^2 + \sum_{i=1}^H \|y_{i-1}\|_2^2 + \lambda \sum_{i=1}^H \|z_{i-1}\|_1,$$

and the constraint set is given by

$$\begin{array}{l} \mathscr{C} = \{(x, y, u, z) |\ x_i = Ax_{i-1} + Bu_{i-1},\ i = 1,\ldots,H \\ \qquad\qquad y_{i-1} = Cx_{i-1} + Du_{i-1},\ i = 1,\ldots,H, \\ \qquad\qquad z_{i-1} = Ex_{i-1} + Fu_{i-1},\ i = 1,\ldots,H\}. \end{array}$$

Thus, the ADMM formulation of optimization problem in (1) is

$$\begin{array}{ll}\text{minimize} & f(x,y,u,z)+I_{\mathscr{C}}(x_c,y_c,u_c,z_c),\\ \text{subject to} & x=x_c,\ y=y_c,\ u=u_c,\ z=z_c.\end{array} \tag{9}$$

B. Step 1 of ADMM

The first ADMM step, (5), is almost the same as the one for ℓ_1 mean filtering in [19]. We solve $4H+1$ separate minimization problems because $f(x,y,u,z)$ is separable in its arguments. For the vector variables x, y and u the minimization problems have a quadratic cost function and no constraints. The solutions are

$$\begin{aligned}
x_i^{k+1} &= x_{c,i}^k - x_{d,i}^k,\ i=0,\ldots,H-1,\\
x_H^{k+1} &= (2Q+\rho I_n)^{-1}\rho(x_{c,H}^k - x_{d,H}^k),\\
y_i^{k+1} &= (2+\rho)^{-1}\rho(y_{c,i}^k - y_{d,i}^k),\ i=0,\ldots,H-1,\\
u_i^{k+1} &= u_{c,i}^k - u_{d,i}^k,\ i=0,\ldots,H-1,
\end{aligned}$$

where I_a denotes the identity matrix in $\mathbf{R}^{a\times a}$. For the vector variable z, the minimization problems are

$$z_i^{k+1} = \underset{z_i}{\arg\min}\{\lambda\|z_i\|_1 + (\rho/2)\|z_i - z_{c,i}^k + z_{d,i}^k\|_2^2\}, \tag{10}$$

with component-wise solutions $(z_i^{k+1})_j = \mathscr{S}_{\lambda/\rho}((z_{c,i}^k - z_{d,i}^k)_j)$, for $i=0,\ldots,H-1$, and $j=1,\ldots,p$, where $\mathscr{S}_{\lambda/\rho}$ denotes the soft thresholder operator, see [7].

C. Step 2 of ADMM

The second step of ADMM, (6), consists of a projection of the vector

$$(x_p^k,y_p^k,u_p^k,z_p^k) = (x^{k+1}+x_d^k, y^{k+1}+y_d^k, u^{k+1}+u_d^k, z^{k+1}+z_d^k)$$

onto the constraint set $\mathscr{C}$, i.e.

$$(x_c^{k+1},y_c^{k+1},u_c^{k+1},z_c^{k+1}) = \Pi_{\mathscr{C}}((x_p^k,y_p^k,u_p^k,z_p^k)).$$

The projection can be formulated as the optimization problem

$$\begin{array}{ll}\text{minimize} & \|(x_c^{k+1},y_c^{k+1},u_c^{k+1},z_c^{k+1}) - (x_p^k,y_p^k,u_p^k,z_p^k)\|_2^2,\\ \text{subject to} & (x_c^{k+1},y_c^{k+1},u_c^{k+1},z_c^{k+1}) \in \mathscr{C}.\end{array} \tag{11}$$

To simplify notation in the rest of this section, we will drop the use of super script k and $k+1$. An equivalent optimization problem to the one in (11) is

$$\begin{array}{ll}\text{minimize} & v^T\mathscr{Q}v + q^Tv\\ \text{subject to} & \mathscr{F}v = g,\end{array} \tag{12}$$

where

$$\begin{aligned}
v &= (x_{c,0},u_{c,0},\ldots,x_{c,H-1},u_{c,H-1},x_{c,H}),\\
g &= (x_{c,0},0,\ldots,0),\\
q &= (r_0,s_0,\ldots,r_{H-1},s_{H-1},0),\\
r_i &= -2(x_{p,i}^T + z_{p,i}^T E + y_{p,i}^T C),\\
s_i &= -2(u_{p,i}^T + z_{p,i}^T F + y_{p,i}^T D),\\
\mathscr{Q} &= \begin{bmatrix} T & 0\\ 0 & Q\end{bmatrix},\quad T = I_H \otimes \begin{bmatrix} P & S\\ S^T & R\end{bmatrix},\\
P &= I_n + C^TC + E^TE,\\
R &= I_l + D^TD + F^TF,\\
S &= C^TD + E^TF,\\
\mathscr{F} &= \begin{bmatrix} I_n & 0 & 0 & 0 & \ldots & 0\\ -A & -B & I_n & 0 & \ldots & 0\\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & \ldots & \ldots & -A & -B & I_n\end{bmatrix}.
\end{aligned}$$

The symbol $\otimes$ denotes the Kronecker product. The optimization problem in (12) is an equality constrained minimization problem. As such, its solution is equivalent to the solution of its Karush-Kuhn-Tucker (KKT) conditions, [1]. The KKT conditions of the optimization problem in (12) are

$$\begin{bmatrix} 2\mathscr{Q} & \mathscr{F}^T\\ \mathscr{F} & 0\end{bmatrix} w - \begin{bmatrix} -q\\ g\end{bmatrix} = 0, \tag{13}$$

with $w=(v,v_d)$, where v_d is the Lagrange multiplier corresponding to the equality constraint $\mathscr{F}v=g$. The KKT conditions in (13) are a system of linear equations and can be efficiently solved using a Riccati recursion as described in the Appendix. The solution to the optimization problem in (11) is obtained by extracting x_c and u_c from v, and calculating y_c and z_c from the equations defining the constraint set $\mathscr{C}$.

D. Step 3 of ADMM

In the third ADMM step in (7), we update the scaled dual variables, i.e.

$$\begin{aligned}
x_{d,i}^{k+1} &= x_{d,i}^k + (x_i^{k+1} - x_{c,i}^{k+1}),\quad i=0,\ldots,H,\\
y_{d,i}^{k+1} &= y_{d,i}^k + (y_i^{k+1} - \tilde{y}_{c,i}^{k+1}),\quad i=0,\ldots,H-1,\\
u_{d,i}^{k+1} &= u_{d,i}^k + (u_i^{k+1} - u_{c,i}^{k+1}),\quad i=0,\ldots,H-1,\\
z_{d,i}^{k+1} &= z_{d,i}^k + (z_i^{k+1} - z_{c,i}^{k+1}),\quad i=0,\ldots,H-1.
\end{aligned}$$

V. Example

In this section, we describe how the ADMM algorithm, proposed in IV, can be used to solve an ℓ_1 regularized MPC problem without inequality constraints.

A. Model

We consider a linear and discrete model of the plant. The model is given by

$$\begin{aligned} x(t+1) &= Ax(t)+Bu(t),\\ y(t) &= Cx(t),\end{aligned} \tag{14}$$

where $x(t)\in\mathbf{R}^n$ is the state vector, $u(t)\in\mathbf{R}^l$ is the input vector and $y(t)\in\mathbf{R}^m$ is the output vector.

B. Cost function

The control objective is to drive the output vector to zero, namely the regulator problem [10], while using a piecewise constant input signal. Such a control objective can be described by the cost function

$$V(t) = \|\hat{x}(t+H_p|t)\|_{2,\bar{Q}}^2 + \sum_{i=1}^{H_p} \|\hat{y}(t+i-1|t)\|_{2,Q}^2 + \lambda \sum_{i=1}^{H_u} \|\Delta\hat{u}(t+i-1|t)\|_1. \quad (15)$$

The cost function penalizes the terminal state, output deviation from zero and non-constant input signals. In (15), $\hat{x}(t+i|t)$ and $\hat{y}(t+i|t)$ are the predicted state and output vectors, respectively, at time $t+i$ given measurements up to time t and the model in (14). Moreover,

$$\Delta\hat{u}(t+i|t) = \hat{u}(t+i|t) - \hat{u}(t+i-1|t),$$

where $\hat{u}(t+i|t)$ is the predicted input vector given measurements up to time t and the model in (14). The prediction and control horizons are denoted H_p and H_u respectively, and we assume that $\Delta\hat{u}(t+i|t) = 0$ for all $i \geq H_u$. The matrices $Q, \bar{Q} \in \mathbf{R}^{m\times m}$ and the scalar λ are weights. We require that Q and $\bar{Q}$ are positive semidefinite, and that λ is non-negative.

C. Optimization problem

The control objective can be achieved by minimizing the cost function in (15) given the model in (14) in each time step t, in accordance with the receding horizon idea, [10]. We can formulate the optimization problem as

$$\begin{array}{ll} \text{minimize} & V(t), \\ \text{subject to} & \hat{x}(t+i|t) = A\hat{x}(t+i-1|t) + B\hat{u}(t+i-1|t), \\ & i = 1,\ldots,H_p, \\ & \hat{x}(t|t) = x(t), \\ & \hat{y}(t+i-1|t) = C\hat{x}(t+i-1|t),\ i = 1,\ldots,H_p. \end{array} \quad (16)$$

The optimization problem in (16) is similar to standard formulations as the one found in [10]. The significant difference is the use of the ℓ_1-norm of $\Delta\hat{u}(t+i|t)$ instead of the ℓ_2-norm in the cost function in (15). The former typically promotes sparse $\Delta\hat{u}(t+i|t)$ for $i = 0,\ldots,H_u-1$, while the latter promotes small but non-zero elements of $\Delta\hat{u}(t+i|t)$ for $i = 0,\ldots,H_u-1$, [1]. To simplify notation in the rest of the paper, we denote $\hat{x}(t+i|t)$ as x_i, $\hat{y}(t+i|t)$ as y_i, and so forth.

D. Receding horizon

The optimization problem in (16) is solved with respect to the input vector u_i for $i = 0,\ldots,H_u-1$. The input vector at the first time step, u_0, is applied to the plant. The state vector is updated according to measurements and, if necessary, an observer. The optimization problem in (16) is updated and solved again. The described procedure is repeated until some final time step. Note that closed loop stability cannot be guaranteed for all values of λ, see [9]. Typically, a terminal cost penalty is used to obtain closed loop stability, if possible.

E. MPC formulation

We consider the optimization problem in (16). We set the predicted output vector to be the predicted state vector, and the prediction horizon equal to the control horizon, that is,

$$\begin{array}{ll} \text{minimize} & \|x_H\|_{2,\bar{Q}}^2 + \sum_{i=1}^{H} \|x_{i-1}\|_{2,Q}^2 + \lambda \sum_{i=1}^{H} \|\Delta u_{i-1}\|_1, \\ \text{subject to} & x_i = Ax_{i-1} + Bu_{i-1},\ i = 1,\ldots,H. \end{array} \quad (17)$$

The optimization problem in (17) can be reformulated by replacing u_i with Δu_i in a similar way as in [10]. We introduce three new vector variables $\tilde{x}_i \in \mathbf{R}^{n+l}$, $\tilde{y}_i \in \mathbf{R}^n$ and $z_i \in \mathbf{R}^l$ in the following way

$$\begin{array}{ll} \text{minimize} & \|\tilde{x}_H\|_{2,\tilde{Q}}^2 + \sum_{i=1}^{H} \|\tilde{y}_{i-1}\|_2^2 + \lambda \sum_{i=1}^{H} \|z_{i-1}\|_1, \\ \text{subject to} & \tilde{x}_i = \tilde{A}\tilde{x}_{i-1} + \tilde{B}\Delta u_{i-1},\ i = 1,\ldots,H, \\ & \tilde{y}_{i-1} = \tilde{C}\tilde{x}_{i-1} + D\Delta u_{i-1},\ i = 1,\ldots,H, \\ & z_{i-1} = E\tilde{x}_{i-1} + F\Delta u_{i-1},\ i = 1,\ldots,H. \end{array} \quad (18)$$

Here, $\tilde{x}_i$ is the state vector augmented with the input vector at the previous time step, i.e. $\tilde{x}_i = (x_i, u_{i-1})$. The matrices $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ are given by

$$\tilde{A} = \begin{bmatrix} A & B \\ 0 & I_l \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B \\ I_l \end{bmatrix}, \quad \tilde{C} = \begin{bmatrix} c & 0 \end{bmatrix},$$

where c is chosen such that $Q = c^T c$, and the matrices D, E, F and $\tilde{Q}$ are given by

$$D = 0, \quad E = 0, \quad F = I_l, \quad \tilde{Q} = \begin{bmatrix} \bar{Q} & 0 \\ 0 & 0 \end{bmatrix}.$$

The optimization problem in (18) is of the same form as the one in (1), with Δu_i acting as the input. Therefore, we can solve (18) efficiently using ADMM and a Riccati recursion as described in Section IV. Note that we in this particular case can pre-calculate F_i, H_i, G_i, and S_i in the Riccati recursion, before we start the MPC iterations.

VI. Numerical examples

In this section, we apply the ADMM algorithm on an ℓ_1 regularized MPC problem. All the examples are performed with $\rho = 1$. To improve convergence we use over-relaxation with $\alpha = 1.8$ and we warm-start each ADMM iteration with the variable values obtained in the previous MPC iteration. We use stopping criteria (8) with $\varepsilon^{abs} = 10^{-5}$ and $\varepsilon^{rel} = 10^{-4}$.

A. Example 1: Quadruple water tank process

1) Plant: The plant is the quadruple water tank process presented in [22]. The process is shown in Figure 1, where $x = (x_1, x_2, x_3, x_4)$ are the water levels, $u = (u_1, u_2)$ are the pump voltages, and $\gamma_1 = \gamma_2 = 0.625$ are the parameters associated with the valves. The area of the cross-sections of the outlets of each tank are (a_1, a_2, a_3, a_4) =(0.17,0.15,0.11,0.08) cm^2, the area of the cross-sections of each tank are $A = 15.5$ cm^2 and the parameters associated with the pumps are $k_1 = k_2 = 4.14$ cm^3/(sV).

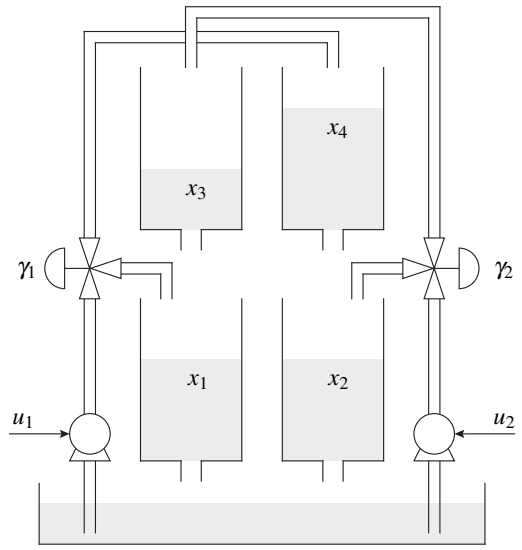

Fig. 1. Quadruple water tank process.

2) Model: We obtain a linear model of the process by linearizing the nonlinear plant description given in [22] around its equilibrium points. The linearized model is

$$\frac{d\bar{x}}{dt} = \begin{bmatrix} \frac{-1}{\tau_1} & 0 & \frac{1}{\tau_3} & 0 \\ 0 & \frac{-1}{\tau_2} & 0 & \frac{1}{\tau_4} \\ 0 & 0 & \frac{-1}{\tau_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{\tau_4} \end{bmatrix} \bar{x}_t + \begin{bmatrix} \frac{\gamma_1 k_1}{A} & 0 \\ 0 & \frac{\gamma_2 k_2}{A} \\ 0 & \frac{(1-\gamma_2)k_2}{A} \\ \frac{(1-\gamma_1)k_1}{A} & 0 \end{bmatrix} \bar{u}_t,$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \bar{x}_t,$$

where $\bar{x} = x - x^0$, $\bar{u} = u - u^0$ and $\tau_i = \frac{A}{a_i}\sqrt{\frac{2x_i^0}{g}}$. The equilibrium points of the plant are $x^0 = (15, 15, 3, 12)$ cm and $u^0 = (7.8, 5.25)$ V. The linear model is discretized assuming zero-order hold sampling at a sampling rate of 1 Hz.

3) Simulation: We set $H = 5$, $Q = I_2$, and $\tilde{Q} = 0$. The plant is initialized with $x(0) = (16, 16, 4, 13)$ cm and $u(0) = u^0$. A Kalman filter is used to estimate the complete state vector during simulation. The MPC iterates for 10 time steps. We perform the same MPC simulation for λ equal to 0.05, 0.1, 2 and 5. The applied input sequences are shown in Figure 2 and the output sequences are shown in Figure 3. We see that the applied input signal varies over time for low values of λ. As λ gets larger, the input signal becomes piece-wise constant, and eventually completely constant. We also see that a more restrictive control strategy, i.e. a high value of λ, gives worse control performance in terms of response time and static error.

B. Example 2: Number of iterations in ADMM

Figure 4 shows the number of iterations required in ADMM for fulfilling the stopping criteria in Example 1. We also investigated the number of iterations required without warm-starting the algorithm. The conclusion is that a warm-start improves the convergence of ADMM when the plant inputs are close to constant and no rapid changes in the plant states occur. When this is not the case, we get similar

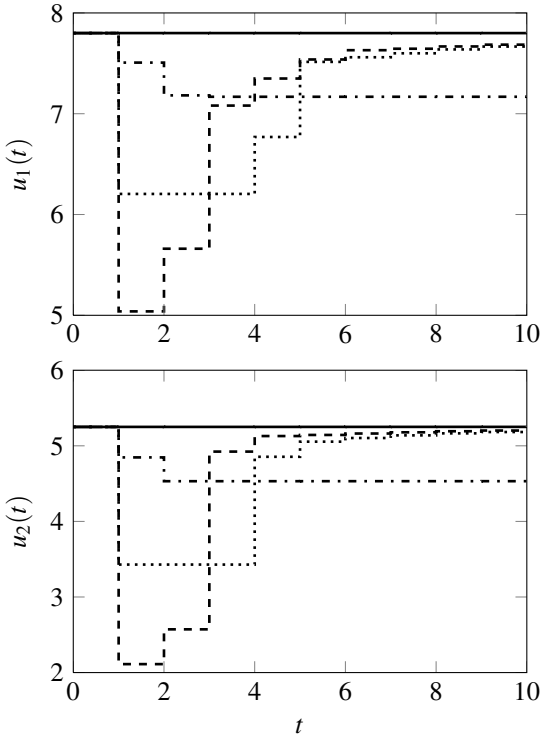

Fig. 2. Applied input sequences. The applied input sequence denoted (- - -), (·····), (· - · -) and (——) corresponds to λ equal to 0.05, 0.1, 2 and 5 respectively.

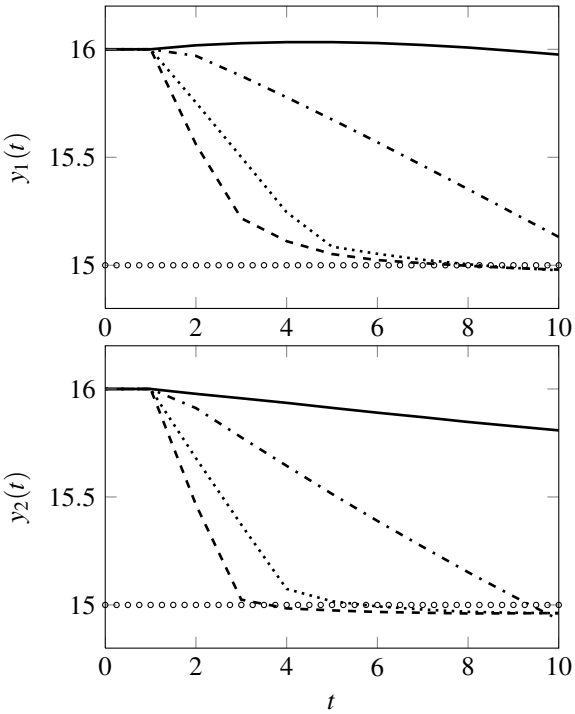

Fig. 3. Measured output sequences. The measured output sequence denoted (- - -), (·····), (· - · -) and (——) corresponds to λ equal to 0.05, 0.1, 2 and 5 respectively. The sequence (∘∘∘) corresponds to the equilibrium point of the water levels.

performance with and without warm-start. It is natural that the benefit of warm-starting is greater the less the states move.

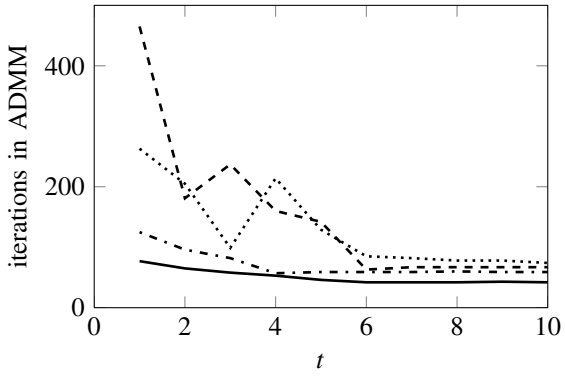

Fig. 4. Number of iterations in ADMM required for fulfilling the stopping criteria. The iteration sequence denoted (- - -), (·····), (· - · -) and (——) corresponds to λ equal to 0.05, 0.1, 2 and 5 respectively. The number of iterations required drops when no rapid changes in plant inputs or states occur.

C. Example 3: Convergence of ADMM

Here we investigate the same set-up as in Example 1. We only consider $\lambda = 0.1$ and the first optimization problem solved in the MPC iterations. We calculate the error of the cost function in (15) for each iteration in ADMM. The error is defined as $e^k = V^* - V^k$, where V^* is the true optimal value of the cost and V^k is the value obtained in ADMM iteration k. The true optimal value is approximated with the solution obtained by running ADMM for 1000 iterations. The optimal value is verified using CVX, a package for specifying and solving convex optimization problems, [23]. CVX calls the generic SDP solvers SeDuMi [24] or SDPT3 [25] to solve the problem. We choose to use SDPT3. The resulting error is shown in Figure 5. The true optimal value is $V^* = 4.58108$, and the final value obtained from ADMM is $V^{264} = 4.58105$, where 264 is the number of iterations required to fulfill the stopping criteria. A rapid drop in the error occur in the first iterations in ADMM. The ADMM algorithm iterates until the stopping criteria are fulfilled, however, for improved visibility of the drop we only show the first 50 iterates. Note that since the ADMM solution is not necessarily feasible it is possible to achieve a value of the cost function at iteration k that is lower than the optimal one. The corresponding primal and dual residuals are shown in Figure 6. We see a rapid drop in error and residuals for the first 20 iterations in ADMM ($e^{20} = -0.03$, $e_p^{20} = 0.11$ and $e_d^{20} = 0.06$), confirming that ADMM converges fast to a moderate accuracy.

D. Example 4: Time of iterations in ADMM

We consider the set-up in Example 1 with $\lambda = 0.1$ and a prediction horizon H varying from 5 to 100 in steps of 5. We only consider the first optimization problem solved in the MPC iterations and we fix the iterations in ADMM to 1000. We calculate the mean value of the time required for an iteration in ADMM. Figure 7 shows the resulting means with respect to the prediction horizon. We see that the mean

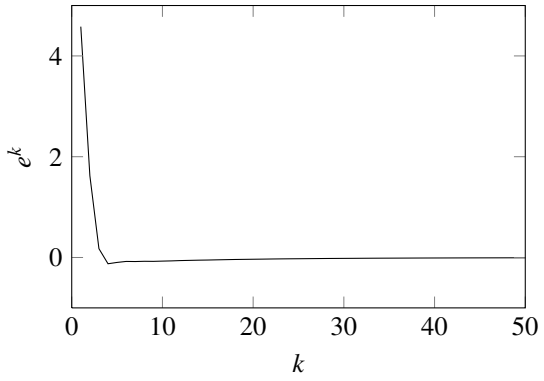

Fig. 5. Error of cost function. The error for each iteration k in ADMM is shown.

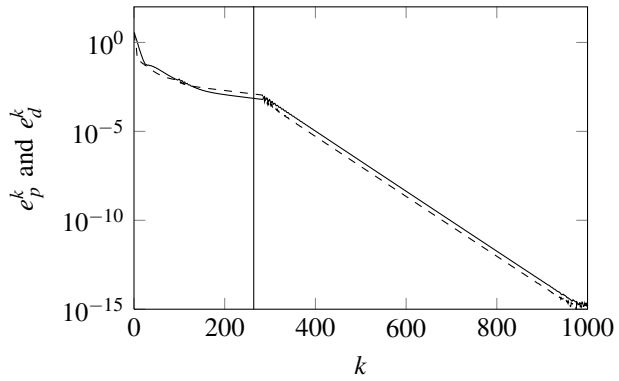

Fig. 6. Primal and dual residuals. The primal (——) and dual (- - -) residuals are calculated for each iteration in ADMM in the first iteration of MPC. The vertical line shows where the stopping criteria are fulfilled. A rapid drop in the residuals occur in the first 20 iterations in ADMM.

time of the iterations in ADMM is linear in the prediction horizon. This is expected since the computational cost of the Riccati recursion is linear in H, [26].

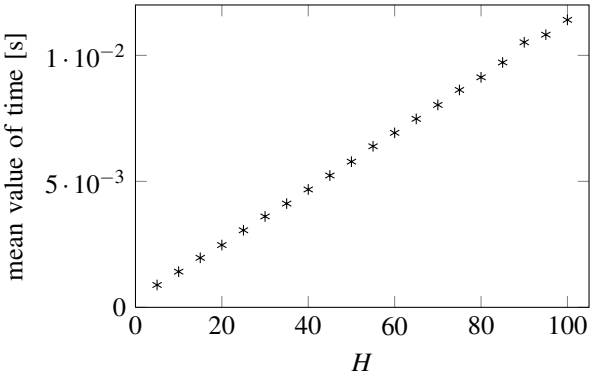

Fig. 7. Mean value of time required for an iteration in ADMM. The mean values are calculated over 1000 iterations in ADMM, for a prediction horizon H varying from 5 to 100 in steps of 5. The mean time is linear in the prediction horizon.

E. Example 5: Required accuracy

Example 3 shows how close the ADMM solution is to the optimal one for $\varepsilon^{abs} = 10^{-5}$ and $\varepsilon^{rel} = 10^{-4}$ in stopping

criteria (8). However, the stopping criteria used may be to conservative with respect to required control performance. For example, if we in Example 1 with $\lambda = 0.1$ restrict the number of iterations in ADMM to 10, we can have a sampling rate of 100 Hz in the MPC, see Figure 7. The input signals obtained with both 10 and 1000 iterations in ADMM are shown in Figure 8. The corresponding output signals are shown in Figure 9. We see that, although the signals differ from each other, they still have the same over-all behavior.

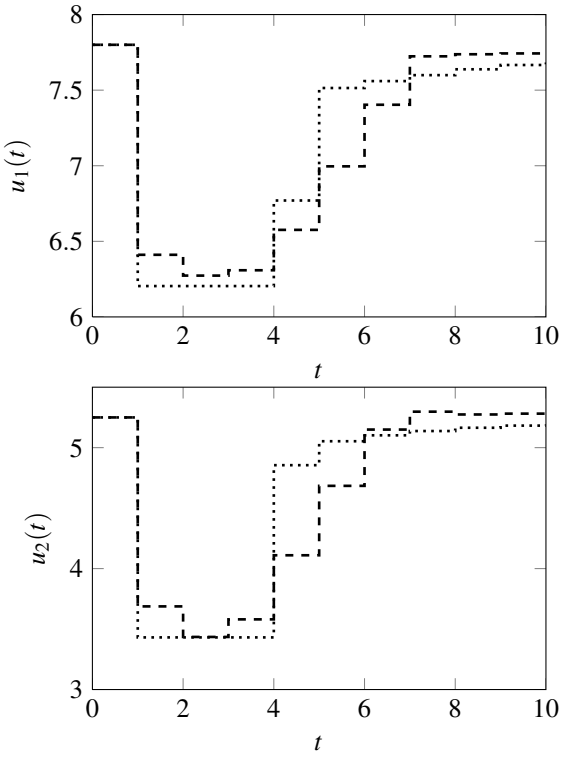

Fig. 8. Applied input sequences. The applied input sequence denoted (- - -) and (·····) corresponds to 10 and 1000 iterations in ADMM respectively. The sequences differ, however the over-all behavior is the same.

VII. Conclusion

We have derived a method for solving optimization problems with an ℓ_1 regularized cost function subject to recursive equality constraints. The optimization problem occurs in control applications, e.g. ℓ_1 regularized MPC. The method is based on the ADMM algorithm. We have showed that the costly projection step in ADMM is equivalent to solving an LQ regulator problem with an additional linear term in the cost function. Such problems can be efficiently solved using Riccati recursion. Future work consists of expanding the proposed method to ℓ_1 regularized cost functions subject to both recursive equality and inequality constraints.

Appendix
Riccati recursion

We use a Riccati recursion to solve the projection problem (6), as in [8]. We showed in Section IV-C that the solution

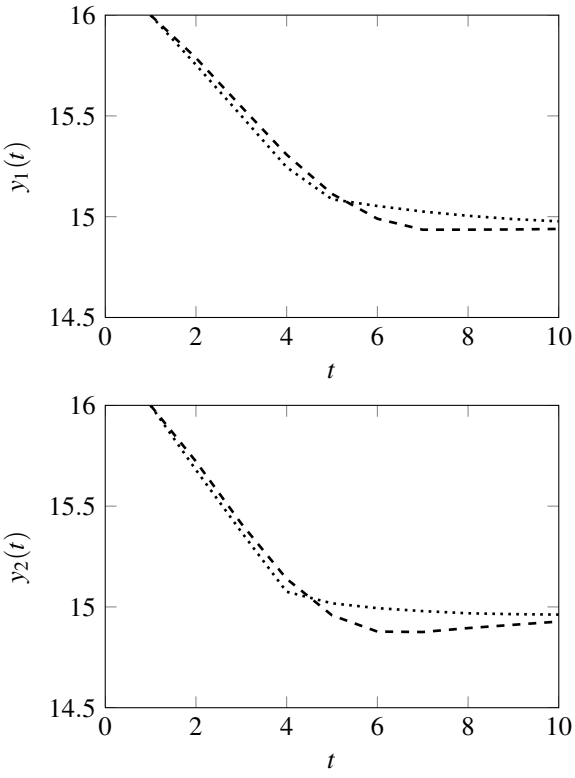

Fig. 9. Measured output sequences. The measured output sequence denoted (- - -) and (·····) corresponds to 10 and 1000 iterations in ADMM respectively. The sequences differ, however the over-all behavior is the same.

to (6) is equivalent to the solution of a system of linear equations,

$$\begin{bmatrix} \mathscr{Q} & \mathscr{A}^T \\ \mathscr{A} & 0 \end{bmatrix} \begin{bmatrix} \xi \\ \lambda \end{bmatrix} = \begin{bmatrix} r_\xi \\ r_\lambda \end{bmatrix}, \tag{19}$$

where $\mathscr{Q}$ and $\mathscr{A}$ are block-diagonal matrices defined as

$$\mathscr{Q} = \begin{bmatrix} Q & 0 \\ 0 & \tilde{Q} \end{bmatrix}, \quad Q = I_H \otimes \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix},$$

and

$$\mathscr{A} = \begin{bmatrix} I & 0 & 0 & 0 & \dots & 0 \\ -A & -B & I & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & -A & -B & I \end{bmatrix}.$$

The vectors ξ, λ, r_ξ and r_λ can be divided into sub-vectors

$$\begin{aligned} \xi &= (x_0, u_0, \dots, u_{H-1}, x_H), & \lambda &= (\lambda_0, \dots, \lambda_H), \\ r_\xi &= (r_{x,0}, r_{u,0}, \dots, r_{u,H-1}, r_{x,H}), & r_\lambda &= (r_{\lambda,0}, \dots, r_{\lambda,H}), \end{aligned}$$

where x_i is given by the system equations

$$x_{i+1} = Ax_i + Bu_i + r_{\lambda,i+1}.$$

It is shown in [26], that there exists a matrix S_i and a vector Ψ_i such that

$$\lambda_i + S_i x_i = \Psi_i, \ i = 0 \dots H,$$

where $S_H = \tilde{Q}$, $\Psi_H = r_{x,H}$, and $x_0 = r_{\lambda,0}$. The matrices S_i and vectors Ψ_i for $i = 0, \dots, H$ can be found through backward

recursion. We then obtain ξ and λ through forward recursion. The algorithm is as follows [26]:
Backward recursion: Update S_i and Ψ_i,

$$\begin{aligned} F_{i+1} &= Q_{11} + A^T S_{i+1} A, \quad H_{i+1} = Q_{12} + A^T S_{i+1} B, \\ G_{i+1} &= Q_{22} + B^T S_{i+1} B, \quad \psi_{i+1} = \Psi_{i+1} - S_{i+1} r_{\lambda,i+1}, \end{aligned}$$

$$\begin{aligned} S_i &= F_{i+1} - H_{i+1} G_{i+1}^{-1} H_{i+1}^T, \\ \Psi_i &= r_{x,i} + A^T \psi_{i+1} - H_{i+1} G_{i+1}^{-1} (r_{u,i} + B^T \psi_{i+1}). \end{aligned}$$

Forward recursion: Update λ_i, u_{i+1} and x_{i+1},

$$\begin{aligned} \lambda_i &= -S_i x_i + \Psi_i, \\ u_{i+1} &= G_{i+1}^{-1} (r_{u,i} + B^T \psi_{i+1} - H_{i+1}^T x_i), \\ x_{i+1} &= A x_i + B u_i + r_{\lambda,i+1}. \end{aligned}$$

A. Unstable model

If A is unstable, the Riccati recursion might not provide the correct solution to (19). To avoid this, we pre-stabilize the state-space equations using state feedback control, see [26], [27], [28]. That is, we let

$$\begin{aligned} x_{i+1} &= A x_i + B u_i, \\ u_i &= -L x_i + v_k, \end{aligned} \tag{20}$$

where L is the feedback vector. We can reformulate (20) as

$$x_{i+1} = (A - BL) x_i + B v_i,$$

and treat v_i as the unknown input signal. The solution obtained from the Riccati recursion will be the values of x_i and v_i. The solution in terms of the original parameters x_i and u_i, can be obtained as

$$\begin{bmatrix} x_i \\ u_i \end{bmatrix} = \begin{bmatrix} I & 0 \\ -L & I \end{bmatrix} \begin{bmatrix} x_i \\ v_i \end{bmatrix}.$$

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. pp. 267–288, 1996.

[3] H. Ohlsson, L. Ljung, and S. Boyd, "Segmentation of arx-models using sum-of-norms regularization," *Automatica*, vol. 46, no. 6, pp. 1107 – 1111, 2010.

[4] J. Eckstein and D. P. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, pp. 293–318, 1992.

[5] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17 – 40, 1976.

[6] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Simul*, vol. 4, pp. 460–489, 2005.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[8] T. Glad and H. Jonson, "A method for state and control constrained linear quadratic control problems," in *Proceedings of the 9th IFAC World Congress*, Budapest, Hungary, 1984.

[9] M. Gallieri and J. M. Maciejowski, "ℓ_{asso} mpc: Smart regulation of over-actuated systems," *To appear in Proceedings of the American Control Conference*, 2012.

[10] J. M. Maciejowski, *Predictive control with constraints*. Prentice Hall, 2002.

[11] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[12] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," Mathematics and Computer Science Division, Argonne National Laboratory, Preprint ANL/MCS-P664-0597, May 1997.

[13] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *Joint 48th IEEE CDC and 28th Chinese Control Conference*, Shanghai, 2009, pp. 7287–7393.

[14] D. Axehill and A. Hansson, "A dual gradient projection quadratic programming algorithm tailored for model predictive control," in *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexiko, Dec. 2008, pp. 3057 – 3064.

[15] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, pp. 816–830, 2008.

[16] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[17] P. Patrinos, P. Sopasakis, and H. Sarimveis, "A global piecewise smooth Newton method for fast large-scale model predictive control," *Automatica*, vol. 47, pp. 2016–2022, 2011.

[18] A. G. Wills, G. Knagge, and B. Ninnes, "Fast linear model predicive control via custom integrated curcuit architecture," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 59–71, 2012.

[19] B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang, "An ADMM algorithm for a class of total variation regularized estimation problems," *To appear in Proceedings of the 16th IFAC Symposium on System Identification*, 2012.

[20] J. Eckstein, "Parallel alternating direction multiplier decomposition of convex programs," *Journal of Optimization Theory and Applications*, vol. 80, pp. 39–62, 1994, 10.1007/BF02196592.

[21] J. Eckstein and M. C. Ferris, "Operator-splitting methods for monotone affine variational inequalities, with a parallel.." *INFORMS Journal on Computing*, vol. 10, no. 2, p. 218, 1998.

[22] K. Johansson, A. Horch, O. Wijk, and A. Hansson, "Teaching multivariable control using the quadruple-tank process," *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, no. December, pp. 807–812, 1999.

[23] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, Apr. 2011.

[24] K. Toh, M. Todd, and R. Tütüncü, "SDPT3—A Matlab software package for semidefinite programming, version 1.3," *Optimization Methods and Software*, vol. 11, no. 1, pp. 545–581, 1999.

[25] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, pp. 625–653, 1999, software available at http://sedumi.ie.lehigh.edu/.

[26] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, pp. 723–757, 1998, 10.1023/A:1021711402723.

[27] S. Keerthi and E. Gilbert, "Optimal infinite-horizon control and the stabilization of linear discrete-time systems: State-control constraints and nonquadratic cost functions," *Automatic Control, IEEE Transactions on*, vol. 31, no. 3, pp. 264 – 266, mar 1986.

[28] J. Rossiter, B. Kouvaritakis, and M. Rice, "A numerically robust state-space approach to stable-predictive control strategies," *Automatica*, vol. 34, no. 1, pp. 65 – 73, 1998.